(Applied) Machine Learning for Data Science

02/6/17

Andreas Müller

Supervised Learning

$$(x_i, y_i) \propto p(x, y)$$ i.i.d.

$$x_i \in \mathbb{R}^n$$

$$y_i \in \mathbb{R}$$

$$f(x_i) \approx y_i$$

Generalization

Not only $$f(x_i) \approx y_i$$

Also for new data: $$f(x) \approx y$$

training set

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix} \qquad y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

test set

Preprocessing

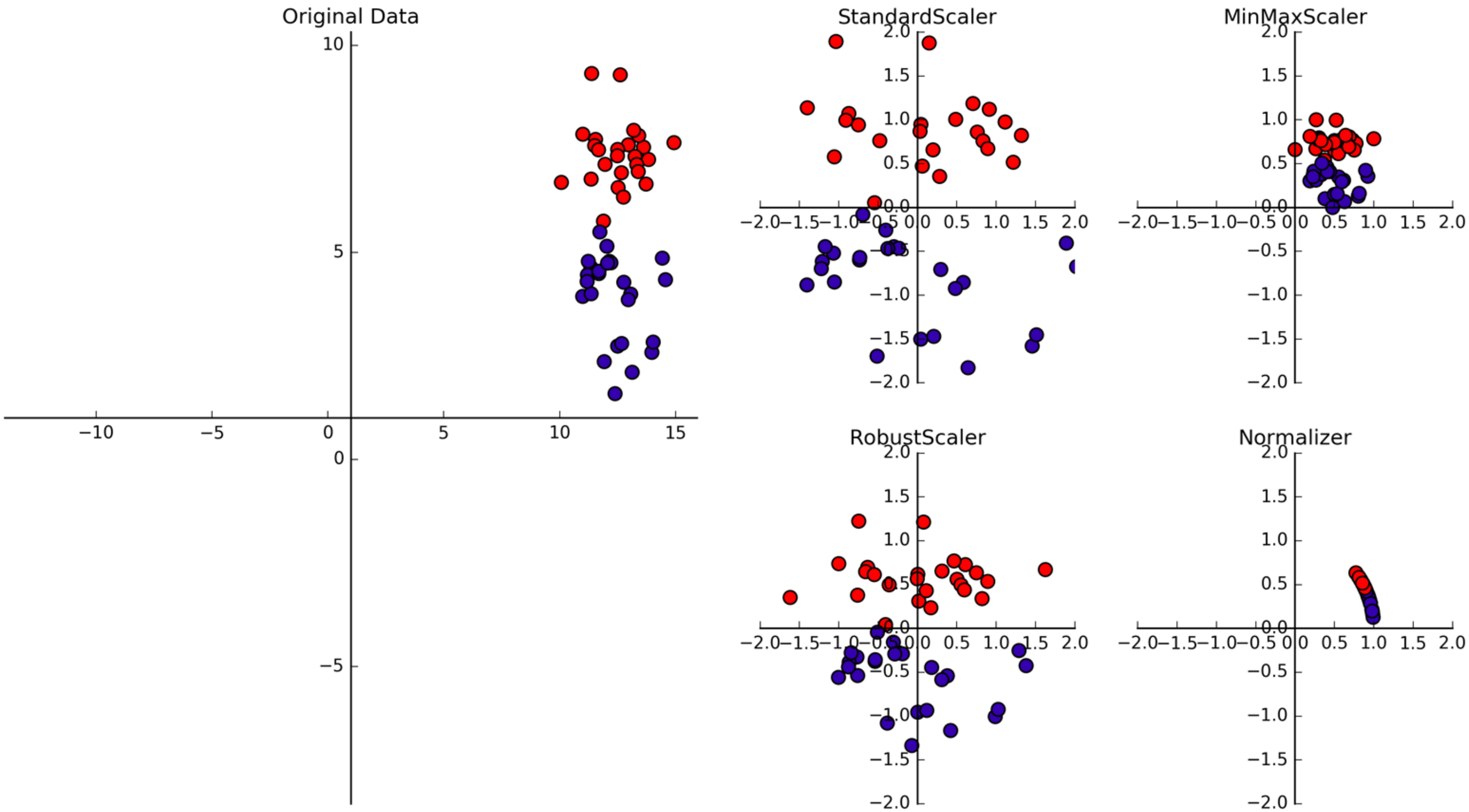
Categorical Variables

$$\mathrm{setup1} \in \mathbb{R}^n ?$$

Categorical Variables

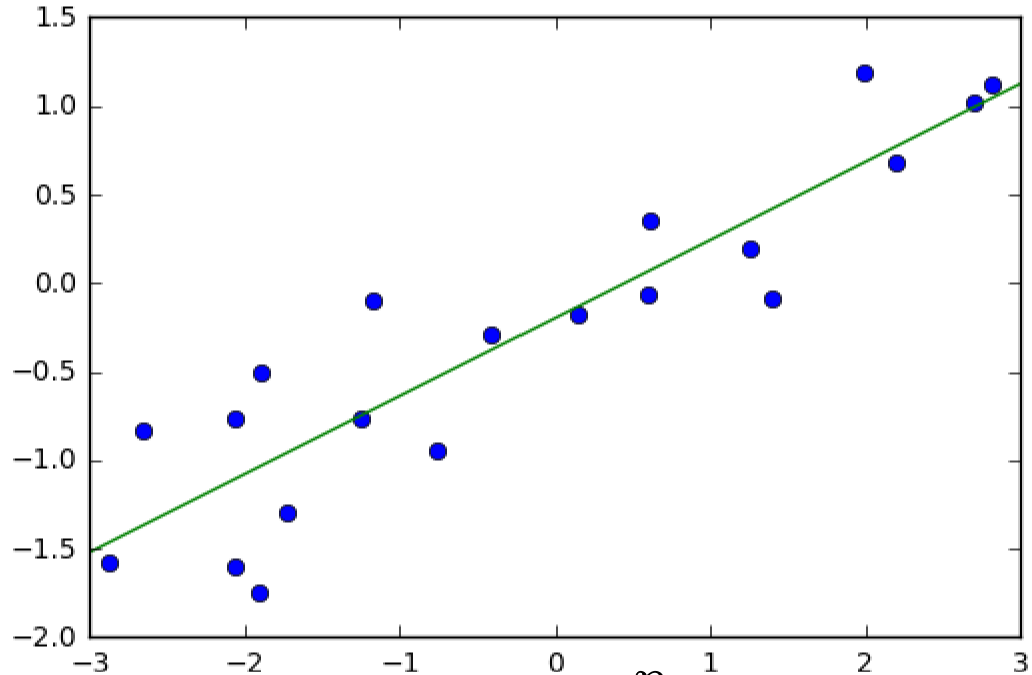
$$\begin{array}{ccc} \text{Setup1} & \text{Setup2} & \text{Setup3} \\ \end{array}$$
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Data Scaling



Linear Models

Linear Regression



$$\hat{y} = w^T \mathbf{x} + b = \sum_{i=1}^{p} w_i x_i + b$$

$$\min_{w \in \mathbb{R}^p} \sum_{i=1}^{n} ||w^T \mathbf{x}_i - y_i||^2$$

Linear Models for Regression

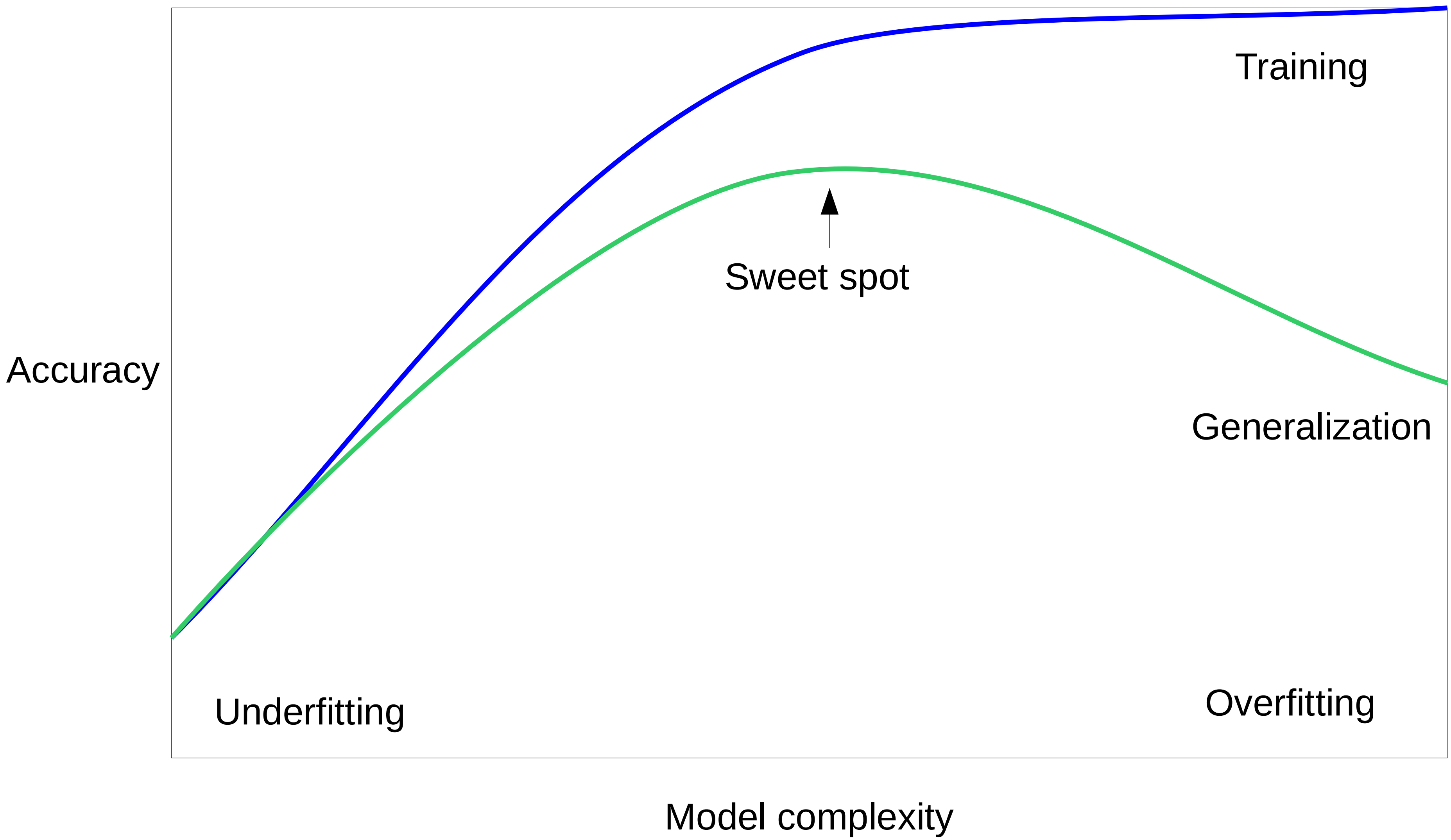
Ridge Regression

$$\min_{w \in \mathbb{R}^n} \sum_i ||w^T x_i - y_i||^2 + \lambda ||w||^2$$

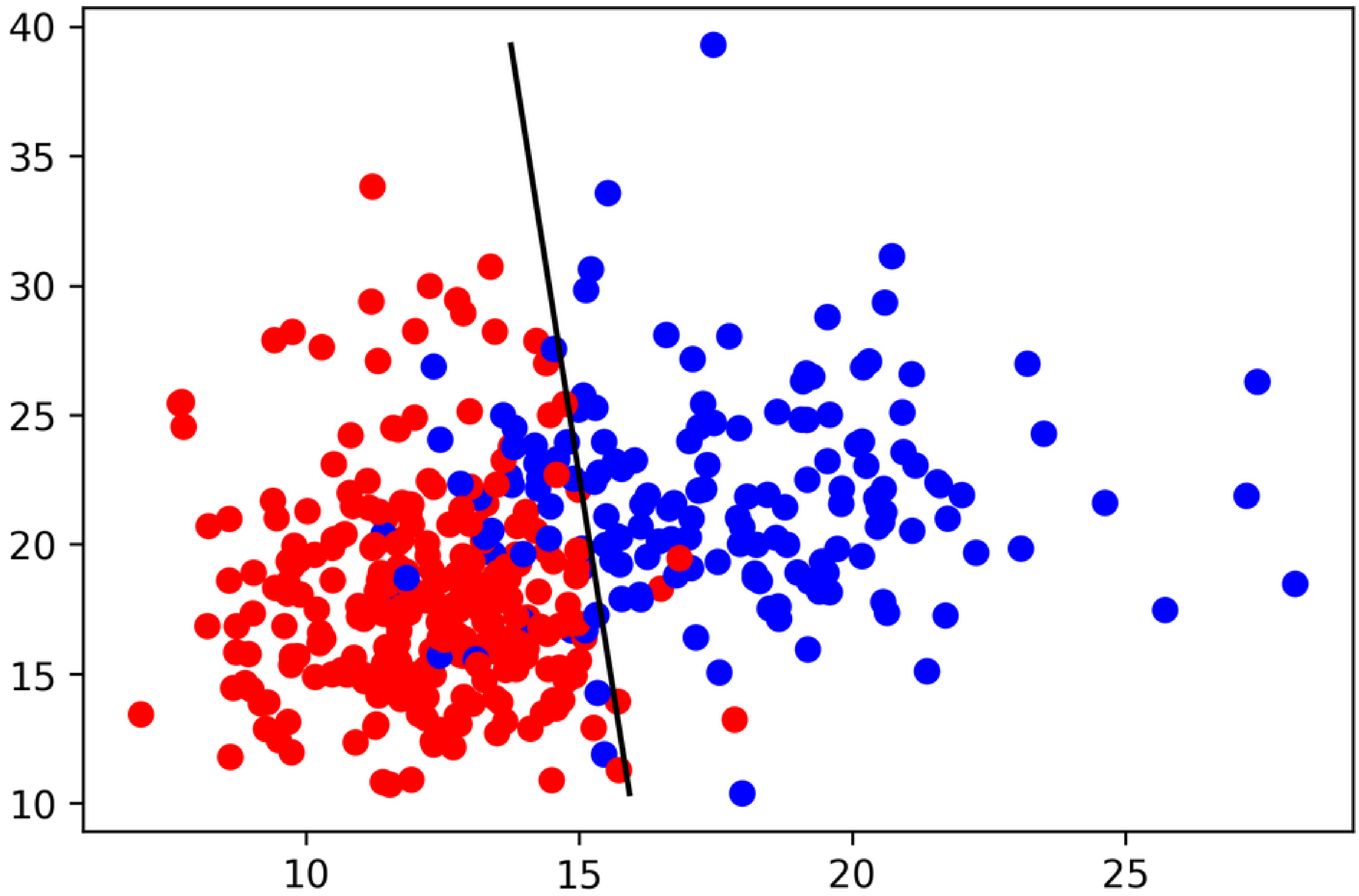
Lasso

$$\min_{w \in \mathbb{R}^n} \sum_i ||w^T x_i - y_i||^2 + \lambda ||w||_1$$

Overfitting and Underfitting



Linear models for **binary** classfiication

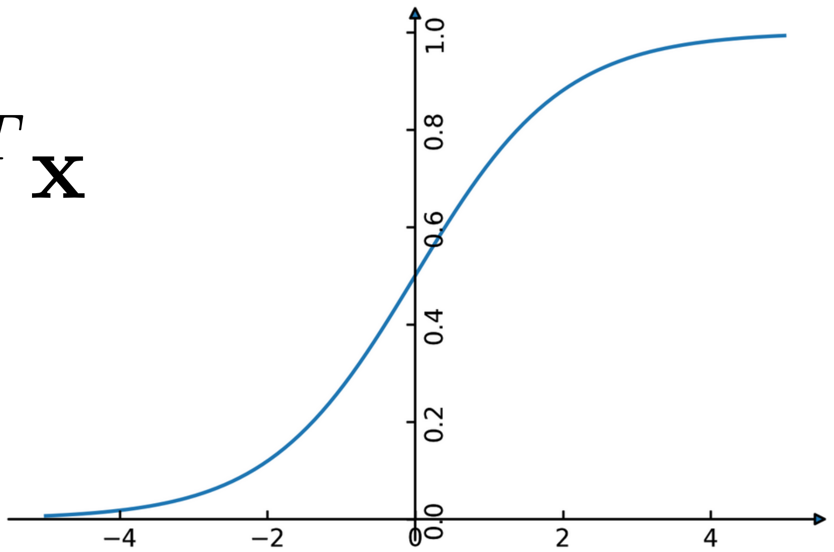


$$\hat{y} = \text{sign}(w^T \mathbf{x} + b) = \text{sign}(\sum_i w_i x_i + b)$$

Logistic Regression

$$\min_{w \in \mathbb{R}^p} -\sum_i \log(\exp(-y_i w^T \mathbf{x}_i) + 1)$$

$$\log\left(\frac{p(y=1|x)}{p(y=0|x)}\right) = w^T \mathbf{x}$$



$$p(y|x) = \frac{1}{1 + e^{-w^T \mathbf{x}}}$$

$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

Penalized Logistic Regression

$$\min_{w \in \mathbb{R}^n} -C \sum_i \log(\exp(-y_i w^T x_i) + 1) + ||w||_2^2$$

$$\min_{w \in \mathbb{R}^n} -C \sum_i \log(\exp(-y_i w^T x_i) + 1) + ||w||_1$$

C is inverse to alpha (or alpha / n_samples)

All points contribute to w (dense solution to dual).

(soft margin) linear SVM

$$\min_{w \in \mathbb{R}^n} C \sum_i \max(0, 1 - y_i w^T \mathbf{x}) + ||w||_2^2$$

$$\min_{w \in \mathbb{R}^n} C \sum_i \max(0, 1 - y_i w^T \mathbf{x}) + ||w||_1$$

Only some points contribute (the support vectors) to w (sparse solution to dual).

SVM or LogReg?

Do you need probability estimates?

no

It doesn’t matter
- try either / both

yes

Logistic Regression

Need compact model or believe solution is sparse? Use L1.

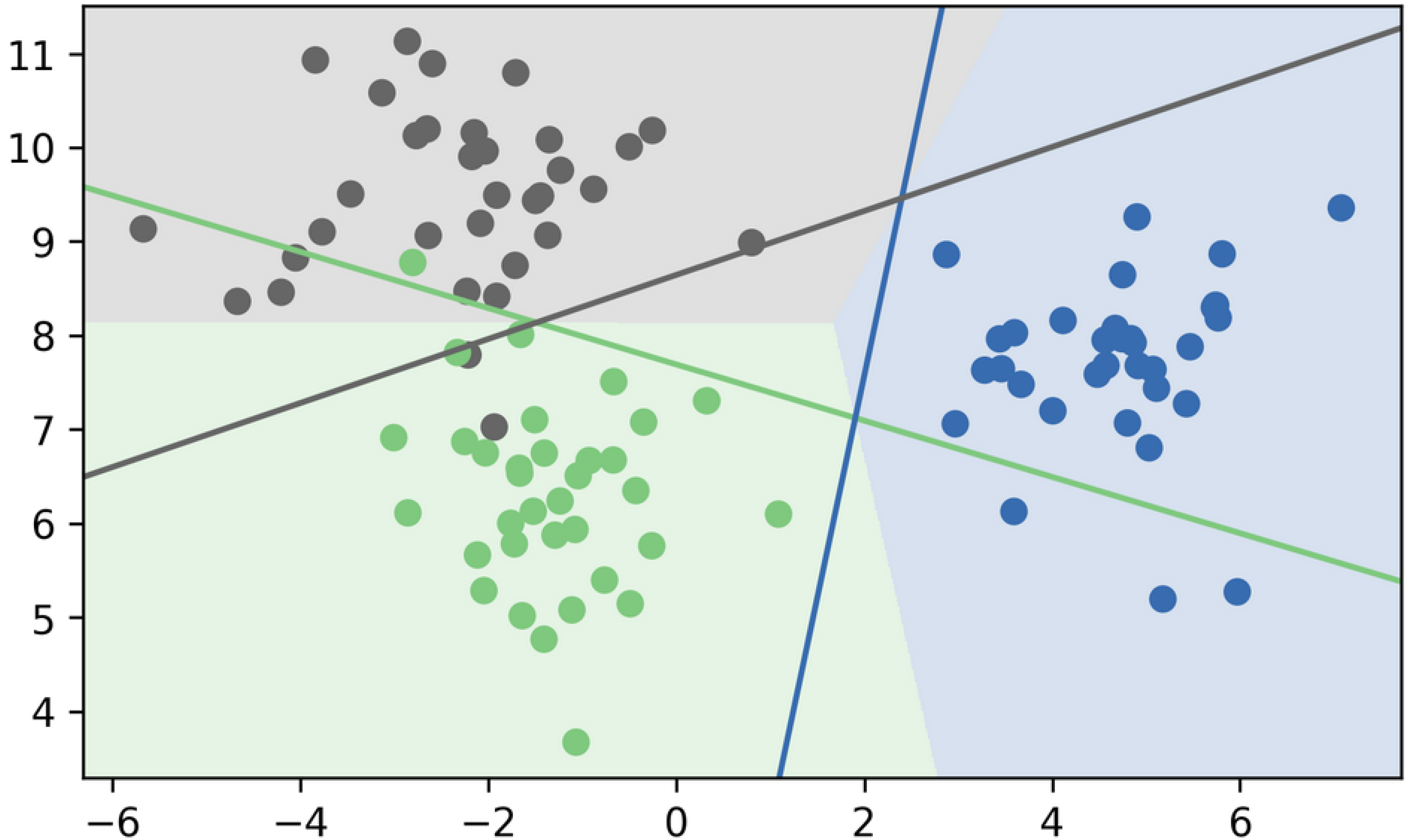
Prediction with One Vs Rest

- "Class with highest score"

$$\hat{y} = \arg\max_{i \in Y} \mathbf{w}_i \mathbf{x}$$

- Unclear why it even works, but work well.

One vs Rest Prediction



Multinomial Logistic Regression

Probabilistic multi-class model:

$$p(y = i|x) = \frac{e^{-\mathbf{w}_i^T\mathbf{x}}}{\sum_{j \in Y} e^{-\mathbf{w}_j^T\mathbf{x}}}$$

$$\min_{w \in \mathbb{R}^n} -\sum_i \log(p(y = y_i|x_i))$$

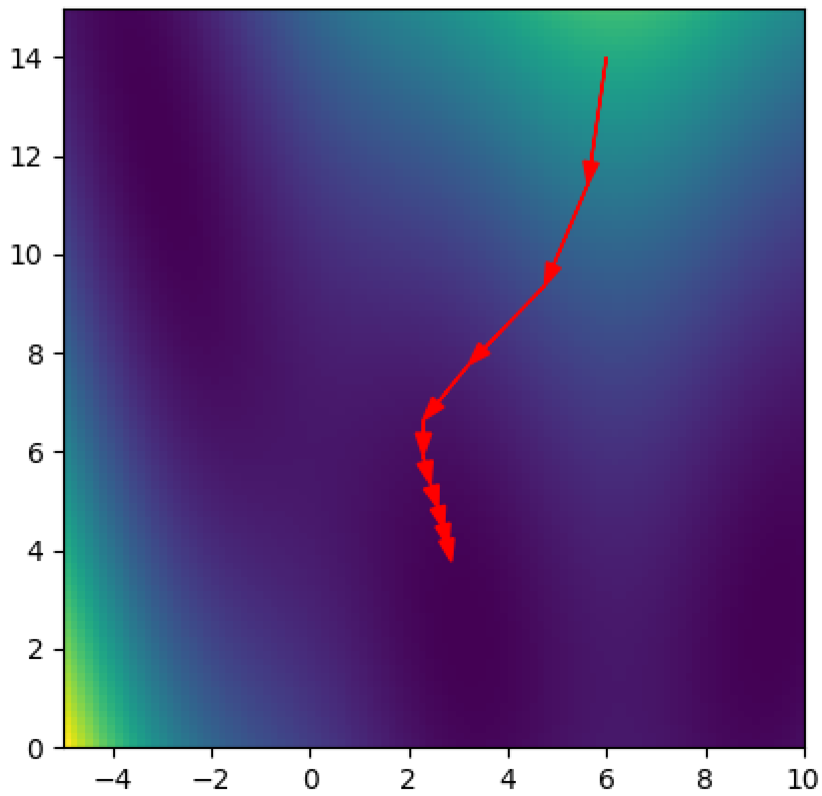
$$\hat{y} = \arg\max_{i \in Y} \mathbf{w}_i\mathbf{x}$$

Same prediction rule as OvR!

Linear models in practice

- Good with P > n : because simple!
  (work well with sparse data)
- Good on **very** large : because fast!
- Very robust, always use as baseline!
- Relatively interpretable
- Try adding interaction or polynomial features

Reminder: Gradient Descent



$$w^{(i+1)} \leftarrow w^{(i)} - \eta_i \frac{d}{dw} F(w^{(i)})$$

Stochastic Gradient Descent

Logistic Regression:

$$F(w) = -C \sum_i \log(\exp(-y_i w^T \mathbf{x}_i) + 1) + ||w||_2^2$$

Sum over data-points

Independent of data

Pick x_i randomly, then

$$\frac{d}{dw} F_i(w) = \frac{d}{dw} - C \log(\exp(-y_i w^T \mathbf{x}_i) + 1) + \frac{1}{n} ||w||_2^2$$

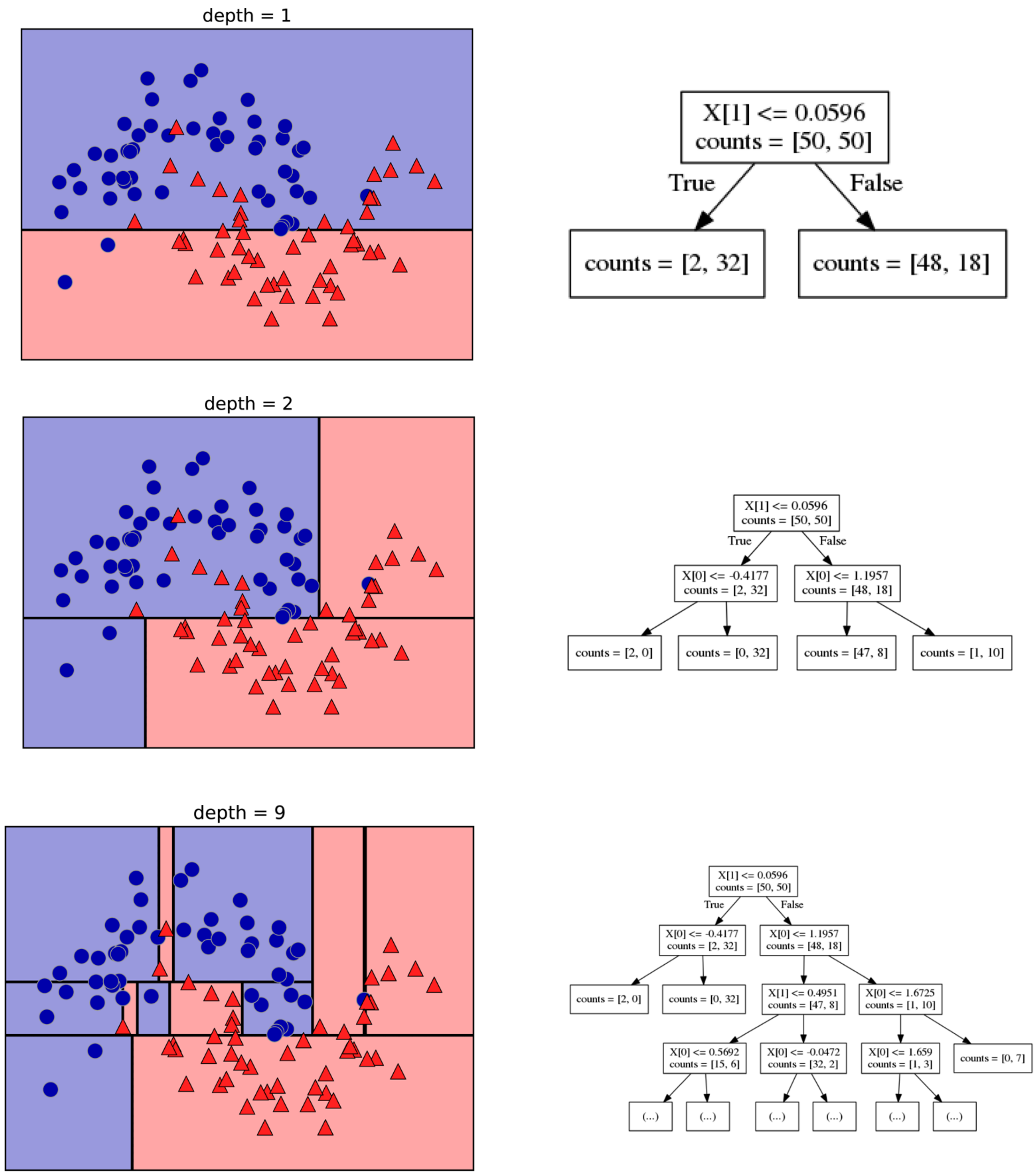
Is a stochastic approximation of gradient of F with expectation the actual gradient.

In practice: just iterate over i.

SGD in Practice

- Vopalwabbit (vw)
- Scikit-learn's SGDClassifier / SGDRegressor
- Can work with data larger than ram
- **VERY** fast
- Picking / tuning learning rates can be hard. Scale data!

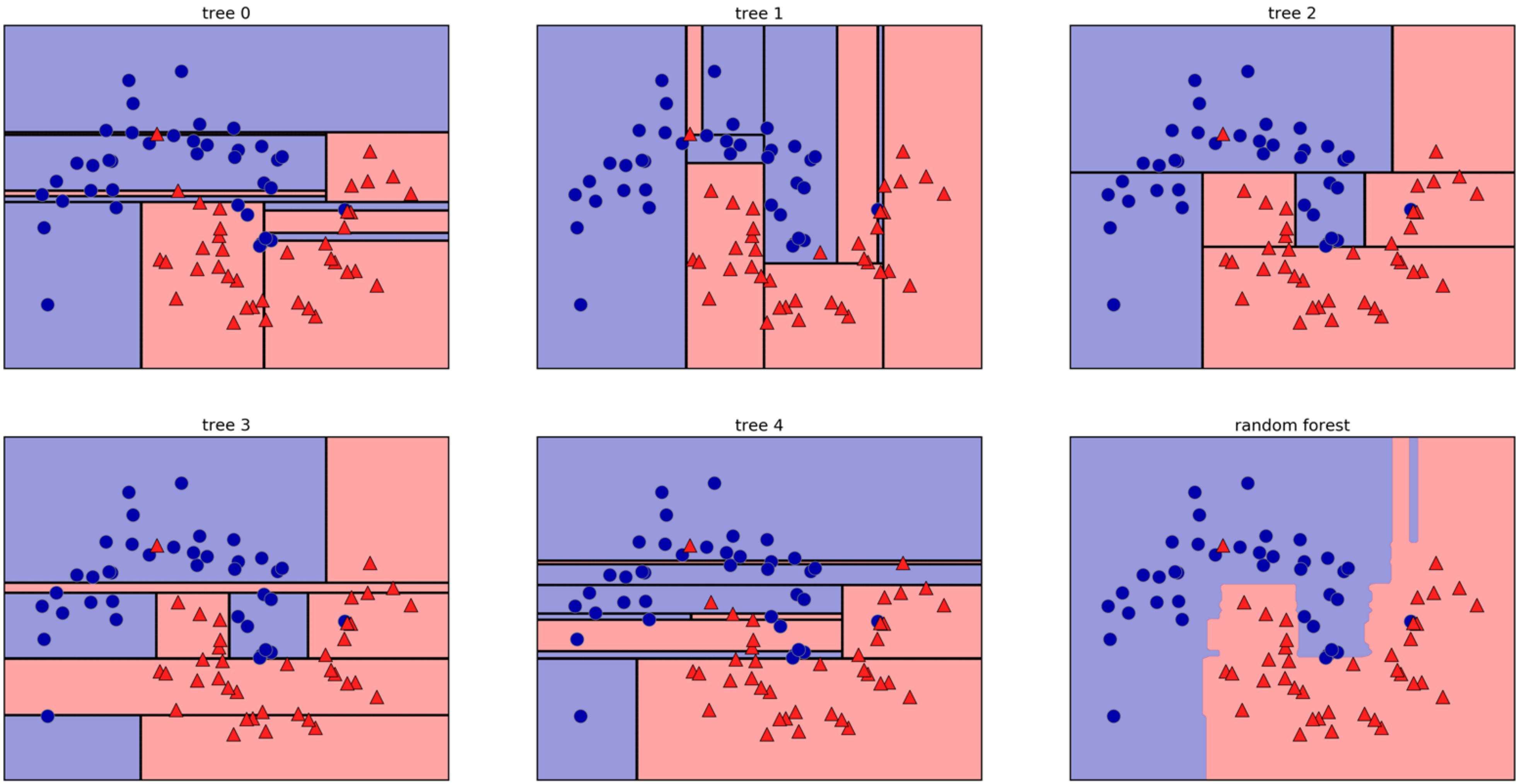
Tree-based models



Decision Trees

Model complexity controlled by depth or number of leafs

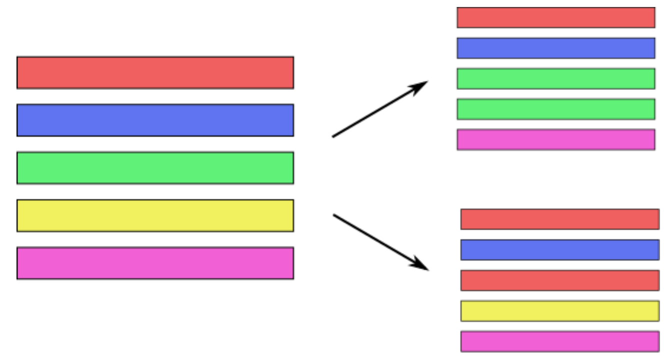
Random Forests



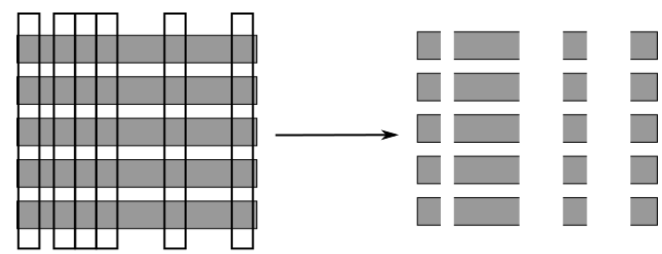
Randomize in two ways

- For each **tree**:

  Pick bootstrap sample of data



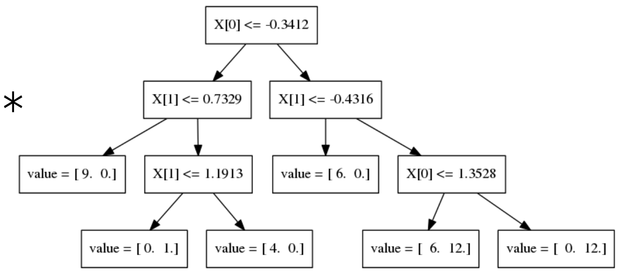
- For each **split**:
  Pick random sample of features



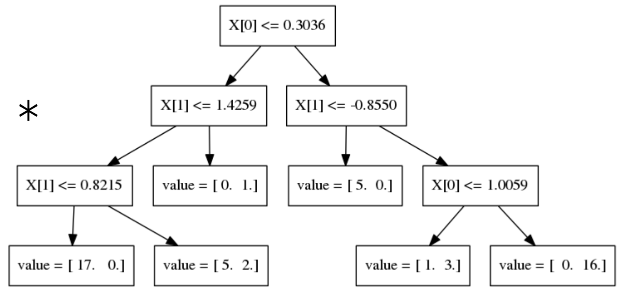
- More tree are always better

Gradient Boosting

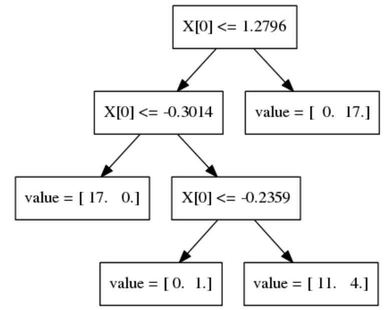
0.1 *



+ 0.1 *



+ 0.1 *

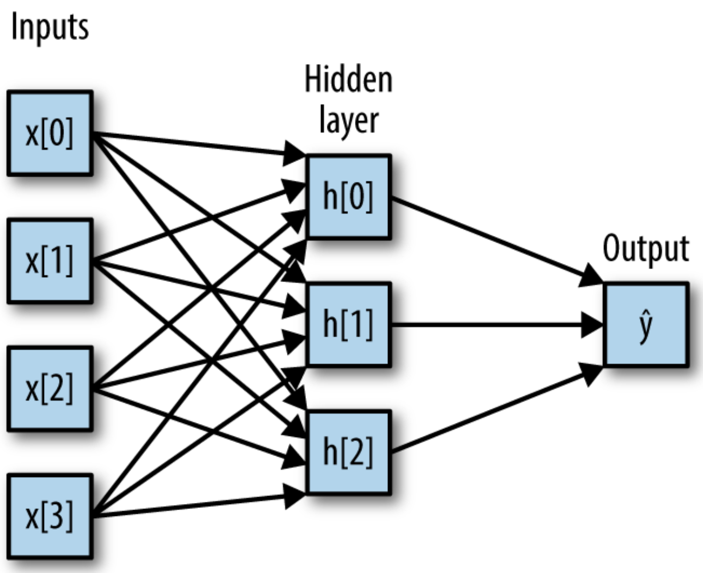


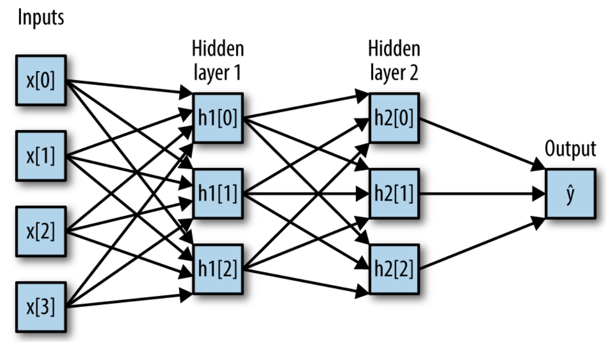
- Many shallow trees
- learning_rate ↔ n_estimators
- Look at XGBoost package

When to use tree-based models

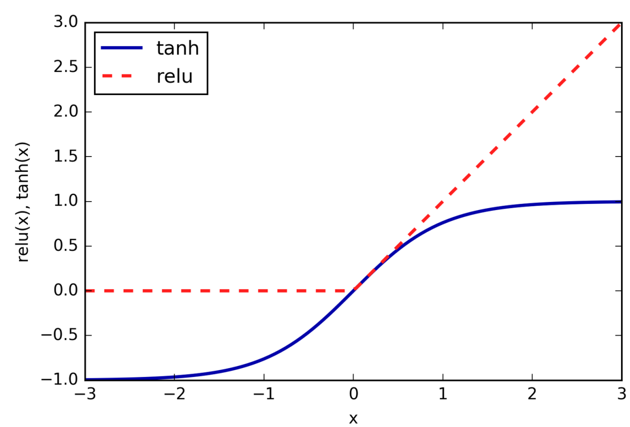
- Model non-linear relationships
- Single tree: very interpretable (if small)
- Random forests very robust, good benchmark
- Gradient boosting often best performance with careful tuning
- Doesn't care about scaling, no need for feature engineering!

Neural Networks





$$\hat{y} = w_2^T \mathrm{relu}(w_1^T x + b_1) + b_2$$



When to use neural networks

- Feed-forward vanilla: lots of data, to get the last couple %.
- Convolutional neural nets: Images, audio, video
- Recurrent neural nets: sequence prediction
- Word-embeddings: text classification

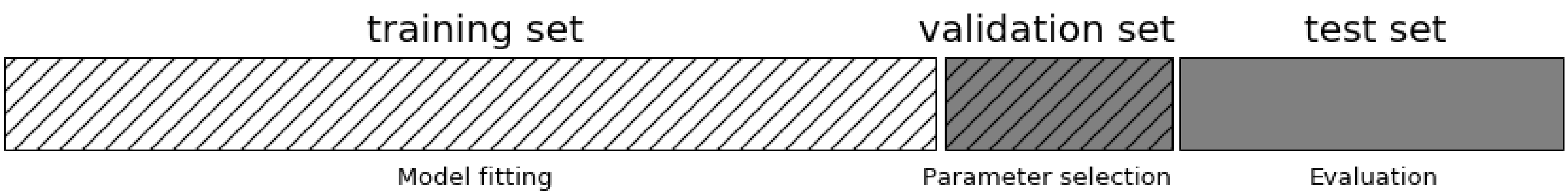
Usually need lots of tuning and preprocessing.

Neural Network / Deep Learning Packages

- Keras (Python)
- Tensorflow (tf.learn) – (mainly Python)
- Caffe (for convolutional nets maybe)
- torch.nn (lua)

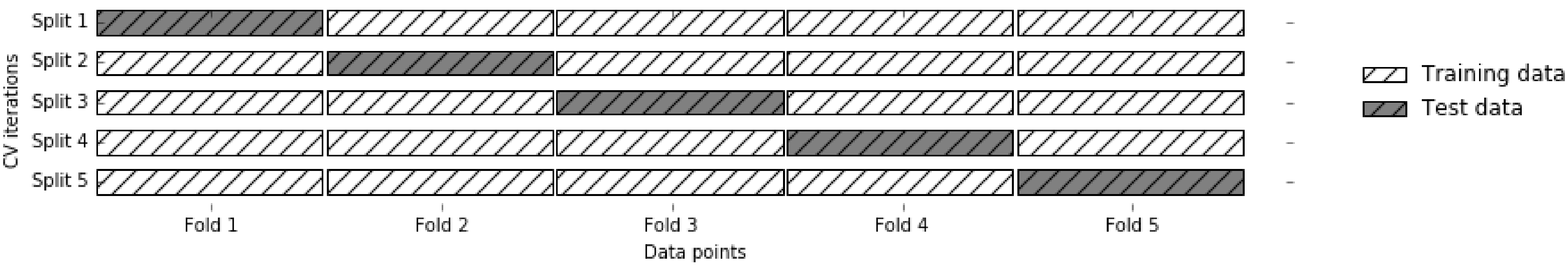
Parameter and Model selection

Three-fold split



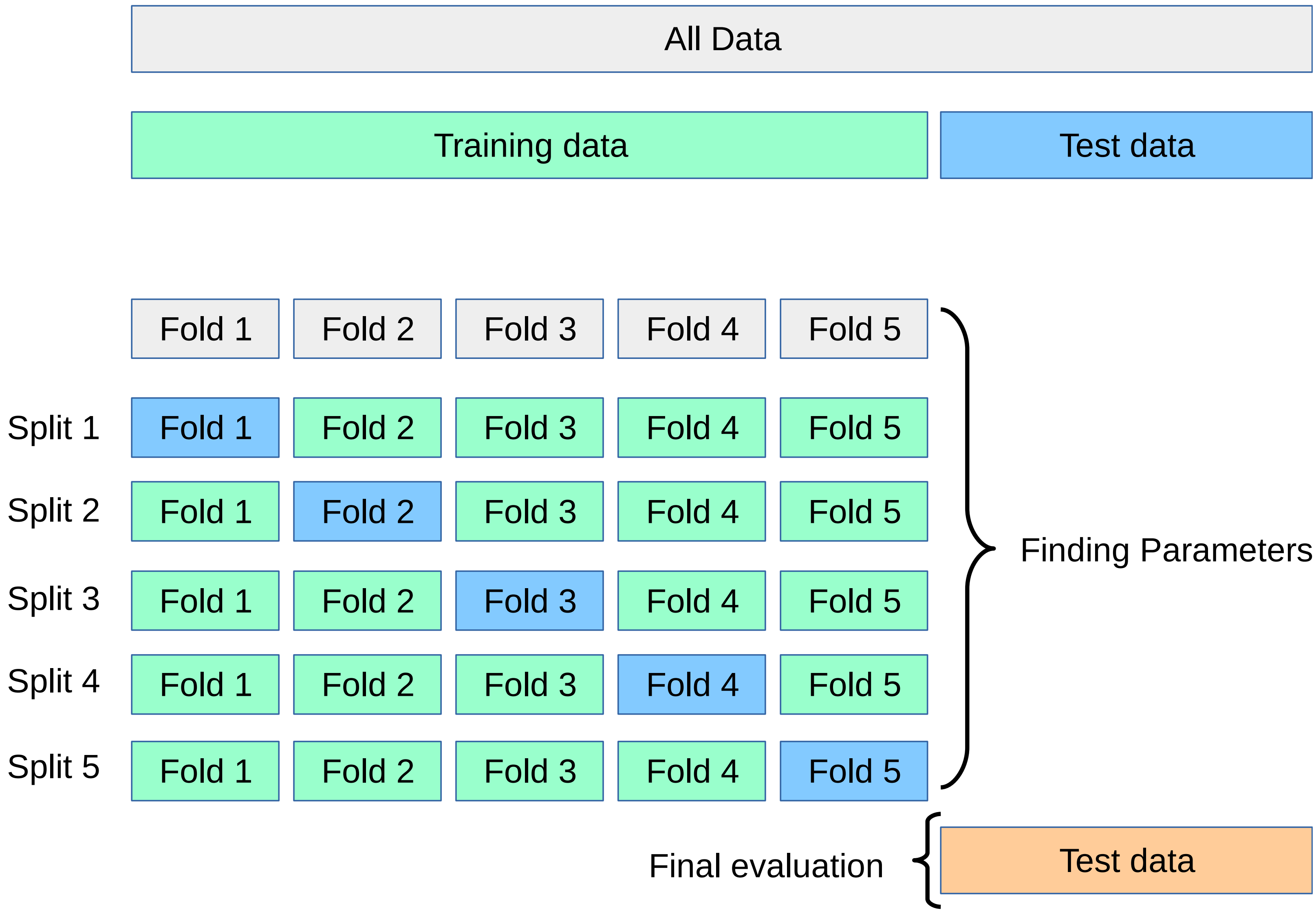
pro: fast, simple
con: high variance, bad use of data.

Cross-validation

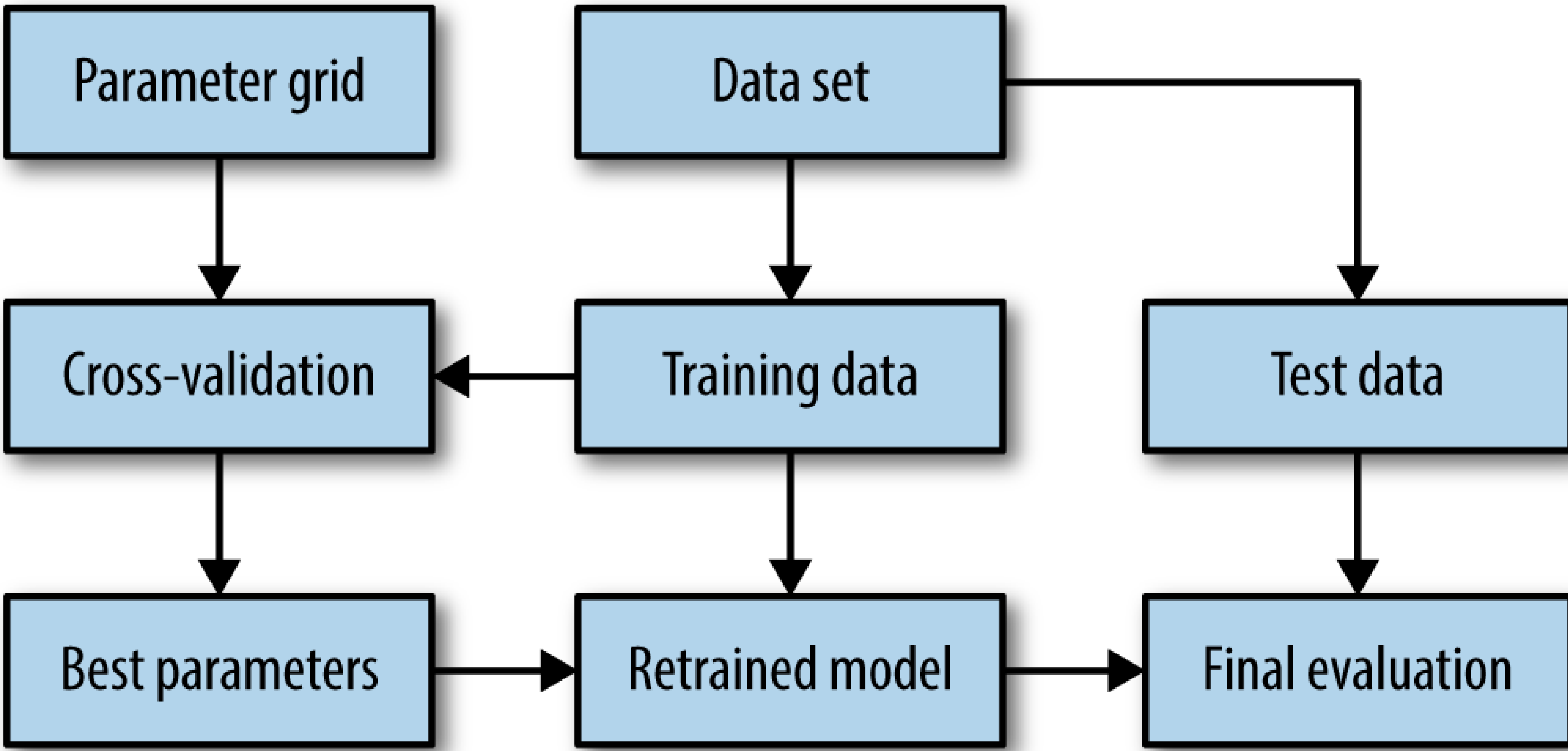


Pro: more stable, more data
con: slower

Cross-validation + test-set



Workflow


Parameter grid
Data set
Cross-validation
Training data
Test data
Best parameters
Retrained model
Final evaluation

Sample application: Sentiment Analysis

see e.g.

https://github.com/amueller/introduction_to_ml_with_python/blob/master/07-working-with-text-data.ipynb

IMDB Movie Reviews Data

Review:
One of the worst movies I've ever rented. Sorry it had one of my favorite actors on it (Travolta) in a nonsense role. In fact, anything made sense in this movie.

Who can say there was true love between Eddy and Maureen? Don't you remember the beginning of the movie ?

Is she so lovely? Ask her daughters. I don't think so.

Label: negative

Training data: 12500 positive, 12500 negative

Bag Of Word Representations

`CountVectorizer / TfidfVectorizer`

`“This is how you get ants.”`

tokenizer

`['this', 'is', 'how', 'you', 'get', 'ants']`

Build a vocabulary over all documents

`['aardvak', 'amsterdam', 'ants', ... 'you', 'your', 'zyxst']`

Sparse matrix encoding

aardvak ants get you zyxst
[0, …, 0, 1, 0, … , 0, 1 , 0, …, 0, 1, 0, …., 0]

N-grams (unigrams and bigrams)

`CountVectorizer / TfidfVectorizer`

`“This is how you get ants.”`

Unigram tokenizer

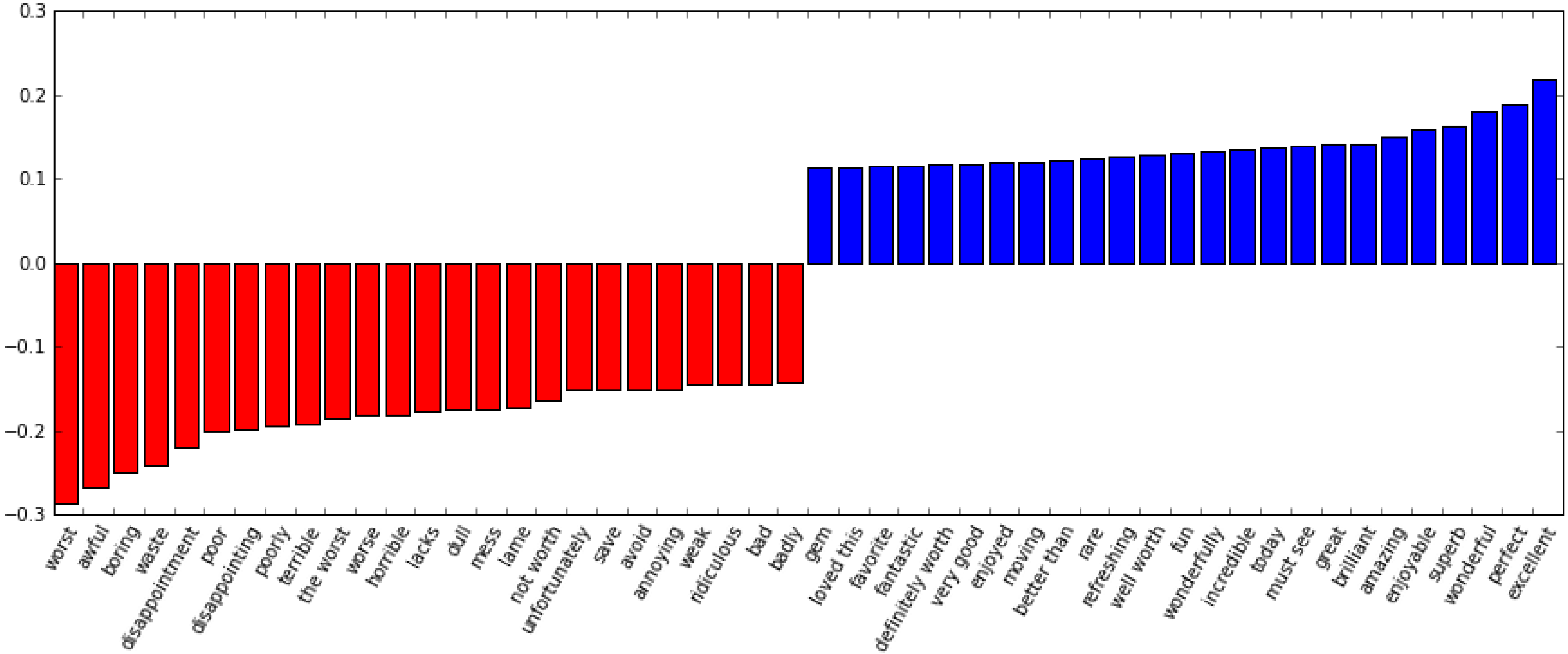
`['this', 'is', 'how', 'you', 'get', 'ants']`

`“This is how you get ants.”`

Bigram tokenizer

`['this is', 'is how', 'how you', 'you get', 'get ants']`

```
text_pipe = make_pipeline(CountVectorizer(), LinearSVC())
text_pipe.fit(text_train, y_train)
text_pipe.score(text_test, y_test)
```



Take-aways

- Try linear models first, random forest second
- Preprocess your data.
- Always keep a hold-out test set
- Do either cross-validation on training set or three-fold split.
- Are you underfitting or overfitting?